TP1 – Estimation de paramètres

Ce TP vise à illustrer l'estimation de paramètres d'une courbe de Bézier de degré variable. Une courbe de Bézier de degré d est définie par ses paramètres, en l'occurrence $d+1$ « points de contrôle ». Dans le cadre de ce TP, nous supposons que ces points de contrôle sont des points du plan $P_i = (\alpha_i, \beta_i)$, $i = 0 \dots d$, dont les abscisses α_i sont uniformément réparties dans l'intervalle $[0,1]$, c'est-à-dire que $\alpha_i = i/d$. En revanche, leurs ordonnées β_i peuvent être librement choisies dans $\mathbb{R}$. Le modèle non bruité de la courbe est défini par :

$$\left\{ \begin{array}{ccc} [0,1] & \rightarrow & \mathbb{R} \\ x & \mapsto & y = f(\beta_0, \beta, \beta_d, x) \end{array} \right.$$

où :

- Le vecteur $\beta = [\beta_1, \dots, \beta_{d-1}]$ a pour coordonnées les paramètres de la courbe autres que β_0 et β_d.
- La fonction $f(\beta_0, \beta, \beta_d, x)$ est définie par :

$$f(\beta_0, \beta, \beta_d, x) = \sum_{i=0}^{d} \beta_i \, B_i^d(x) \tag{1}$$

- Les « polynômes de Bernstein » $B_i^d(x)$, $i = 0 \dots d$, sont définis par :

$$B_i^d(x) = \mathrm{C}_i^d \, x^i (1-x)^{d-i} \tag{2}$$

  où C_i^d est le nombre de combinaisons de i objets parmi d, calculable par la fonction `nchoosek` de Matlab.

On remarque que :

$$\left\{ \begin{array}{l} f(\beta_0, \beta, \beta_d, 0) = \beta_0 \\ f(\beta_0, \beta, \beta_d, 1) = \beta_d \end{array} \right.$$

ce qui signifie que le premier point de contrôle $P_0 = (\alpha_0, \beta_0)$ et le dernier point de contrôle $P_d = (\alpha_d, \beta_d)$ se trouvent sur la courbe de Bézier. En revanche, les autres points de contrôle ne se trouvent généralement pas sur cette courbe : ils jouent seulement le rôle d'« attracteurs ».

Exercice 1 : création des données d'apprentissage

Les données d'apprentissage sont produites à partir d'une courbe de Bézier de degré 5, dont les paramètres sont notés β_0^*, $\beta^* = [\beta_1^*, \beta_2^*, \beta_3^*, \beta_4^*]$ et β_5^*, bruitée par un bruit blanc additif gaussien d'écart-type $\sigma = 0,5$. Ces données d'apprentissage constituent un ensemble $\mathcal{D}_{\text{app}} = \{(x_j, y_j), j = 1 \dots n_{\text{app}}\}$ de n_{app} points du plan, dont les abscisses x_j sont uniformément réparties dans l'intervalle $[0,1]$ et dont les ordonnées y_j sont définies par :

$$y_j = f(\beta_0^*, \beta^*, \beta_5^*, x_j) + b_j \tag{3}$$

où :

$$b_j \sim \mathcal{N}(0, \sigma^2) \tag{4}$$

Écrivez la fonction `bezier`, appelée par le script `exercice_1`, qui calcule la valeur de l'ordonnée d'une courbe de Bézier selon le modèle (1)+(2). Cette fonction devant être valide quelle que soit la dimension du vecteur `x`, il convient d'utiliser les opérateurs `.^` et `.*` de Matlab.

Écrivez la fonction `bezier_bruitee`, appelée par le script `exercice_1`, qui permet de produire des données d'apprentissage selon le modèle (3)+(4). Pour cela, il convient d'utiliser la fonction `randn` de Matlab.

Exercice 2 : estimation des paramètres d'une courbe de Bézier

On souhaite estimer les paramètres d'un modèle à partir des données d'apprentissage $\mathcal{D}_{\text{app}}$. Le modèle retenu est à nouveau une courbe de Bézier. On suppose connus les paramètres β_0^* et β_5^*. Les paramètres inconnus sont donc le degré d de la courbe de Bézier et le vecteur $\beta = [\beta_1, \ldots, \beta_{d-1}]$. Ce problème d'estimation étant linéaire vis-à-vis de β, mais non linéaire vis-à-vis de d, il semble raisonnable de commencer par faire une hypothèse sur la valeur de d, et d'estimer β au sens des moindres carrés ordinaires.

Établissez (sur papier) les expressions de la matrice $\mathbf{A}$ et du vecteur $\mathbf{B}$ du système linéaire suivant, qui traduit le fait que la courbe de Bézier recherchée doit passer exactement par les n_{app} points (x_j, y_j) :

$$\mathbf{A}\,\beta^\top = \mathbf{B} \tag{5}$$

Écrivez la fonction `moindres_carres`, appelée par le script `exercice_2`, censée retourner la solution approchée $\widehat{\beta}$ de l'équation (5), au sens des moindres carrés ordinaires. Pour cela, utilisez soit l'opérateur \ de Matlab, soit la fonction `pinv` (« pseudo-inverse »).

Testez différentes valeurs du degré d entre 2 et 20. Pour les valeurs élevées de d, comprenez-vous ce que signifie le concept de « sur-apprentissage » ?

Exercice 3 : calcul de l'erreur d'apprentissage

L'erreur d'apprentissage (ou *risque empirique*) est définie comme l'écart quadratique moyen entre les données d'apprentissage y_j et les prédictions obtenues à partir du modèle appris $f(\beta_0^*, \widehat{\beta}, \beta_5^*, x_j)$.

Écrivez la fonction `erreur_apprentissage`, appelée par le script `exercice_3`, qui calcule l'erreur d'apprentissage pour la valeur du degré d passée en paramètre. Vous semble-t-il possible d'estimer le degré du modèle exact ($d = 5$) à partir de cette courbe ?

Exercice 4 : absence de biais de l'estimation (question facultative)

Dans cette question, on suppose que d et connu et égal à 5.

On peut montrer que le vecteur de paramètres $\widehat{\beta}$ estimé par les moindres carrés ordinaires est distribué selon la loi normale suivante, lorsque les données d'apprentissage $\mathcal{D}_{\text{app}}$ varient :

$$\widehat{\beta} \sim \mathcal{N}(\beta^*, \sigma^2\,(\mathbf{A}^\top\mathbf{A})^{-1})$$

Cela montre que l'estimation de paramètres, telle qu'elle a été effectuée, est non biaisée.

Écrivez un script de nom `exercice_4`, qui consiste à tirer un grand nombre de données d'apprentissage (cf. exercice 1), et qui moyenne les différents vecteurs de paramètres $\widehat{\beta}$ estimés (cf. exercice 2), afin de vérifier l'absence effective de biais.